JOURNAL OF COMPUTER AND SYSTEM SCIENCES 22, 98–105 (1981)

An Improved Simulation Result for Ink-Bounded Turing Machines

ROBERT MELVILLE

Department of Computer Science, Cornell University, Ithaca, New York 14853

Received March 1, 1979

A (one tape, deterministic) Turing Machine is $f(n)$ ink bounded if the machine changes a symbol of its work tape at most $f(n)$ times while processing an input of length n. The result of our paper is the construction of an "ink efficient" universal machine which, for any $f(n)$ ink bounded machine M and input x, can simulate the processing of M on x or detect that M is looping infinitely on input x. The universal machine requires $O(f(n)^{1+\varepsilon})$ ink for this simulation, where ε is an arbitrarily small positive number. As a corollary we establish that for ink-constructable g: For any $\varepsilon > 0$, if $\inf_{n\to\infty}(f(n)^{1+\varepsilon}/g(n)) = 0$ then there is a language in INK$(g(n))$ not in INK$(f(n))$.

1. Problem Statement and Preliminary Lemmas

Let M be a one tape deterministic Turing machine. We say that M is $f(n)$ ink bounded if M changes a symbol of its work tape at most $f(n)$ times while processing any input of length n. If M is scanning its work tape and changing state, but not changing symbols on its tape, we say it is silent. Note that an ink-bounded machine could still diverge if it is caught in an infinite silent loop.

Reference [4] discusses INK complexity and demonstrates that this measure is ill defined for a machine with two or more tapes if the initial contents of the tapes are all blanks. The machine could mark a zero point on both tapes then use the head positions as counters to store an arbitrarily large integer and so compute any recursive function with only finite ink. Our discussion is limited to single tape machines simulated by a single tape machine. Moreover, we make no assumptions about the initial contents of the tape of the universal machine.

We want to design a universal machine U which given any M, as above, and input x can simulate the computation of $M(x)$ or detect that M is diverging. U is to use as little ink as possible.

Next we give three simple lemmas which are used later in the efficiency analysis of U. We suppose M to have a one-way infinite tape and r states.

LEMMA. *If M scans silently more than r cells past the right end of the nonblank portion of its work tape, it will continue to move right forever.*

0022-0000/81/010098–08$02.00/0

Proof. M must repeat a state while scanning a blank and moving right.

LEMMA. *M can form nonblank tape contents of length at most $n + rf(n)$.*

Proof. This follows from the previous lemma since M can move at most r steps to the right without printing.

LEMMA. *If M moves silently more than $r(n + rf(n) + r)$ steps it is looping.*

Proof. M must keep its head within the first $n + rf(n) + r$ cells of the tape. If M repeats one if its r states at the same tape cell it must loop since the tape remains unchanged.

2. THE SIMULATION

Roughly, if U is to detect M looping, U must be able to count $O(f(n))$ silent moves of M. This suggests a straightforward quadratic simulation. U acts like any universal machine and also maintains a count (in unary) of the number of silent moves of M last printed. If this count ever exceeds $r(n + rf(n) + r)$, U knows that M is looping silently. U must spend $O(1)$ ink to simulate one move of M, whether or not M printed on that move, since U must also count. In the worst case, M might move silently $r(t + r) - 1$ steps between printing where t is the length of the non-blank portion of M's work tape. Since t can become $O(f(n))$ (at most) and M may print $O(f(n))$ times, this processing may require $O(f(n)^2)$ ink by U.

Two observations lead to an improved simulation:

(1) If M is scanning silently back and forth over a small portion of the tape, it will soon loop.

(2) If M scans silently across most of the tape, then prints, the bulk of the tape is unchanged and M would repeat the same silent computation on the unchanged portion of the tape.

Let M have states $q_1, q_2, ..., q_r$. Given a tape segment of length s and M in state q_i scanning the leftmost symbol of the segment there are five possibilities:

(1) M scans the segment silently and exits on the right in some state q_i'.

(2) as in (1) but exit on the left,

(3) M loops silently within the segment,

(4) M prints within the segment,

(5) M halts within the segment.

There are the same five possibilities if M starts at the right of the segment.

For any state q_i and tape segment of length s we can determine which of these five cases occurs using at most $O(s)$ ink. This follows from a previous lemma since if M

moves more than rs steps without printing, it must be looping silently within the segment.

Given a segment we can compute the left-entry (respectively right-entry) segment table which records, for each state q_i, which of the five possibilities above occurs, assuming M enters the segment from the left (respectively right). These tables are written in space proportional to r and are computed using ink proportional to the length of the segment.

We demonstrate the simulation for a bound of $O(f(n)^{4/3})$. This will suffice to illustrate all the major points of our construction. The simulated work tape of M is partitioned into segments and each segment is delimited by a left-entry and a right-entry segment table. Furthermore, to achieve the claimed bound, this partitioning is applied recursively once again so that segments are partitioned into sub-segments also delimited by segment tables. We refer to the outer segments as "big" segments or "level 1" segments and the inner segments as "little" or "level 2" segments.

Figure 1 shows a possible tape layout for U. Here $t \leqslant O(f(n))$ is the length of the non-blank portion of M's work tape so far in the simulation. There are $t^{1/3}$ big segments each of length $t^{2/3}$. Each big segment is divided into $t^{1/3}$ little segments of length $t^{1/3}$ each.

Track 1 holds the simulated tape contents and the segment tables for levels 1 and 2.

Track 2 stores the current state, q, of the simulated machine written in the tape alphabet of U. The length of this state description is some constant "$q$size" depending on the number of states in M and the size of the tape alphabet of U. The universal machine will keep this state description under the segment table currently being accessed.

Track 3 holds a counter written in unary notation. This counter is incremented for every silent segment transition made by M since the last printing move and is used to detect infinite silent looping.

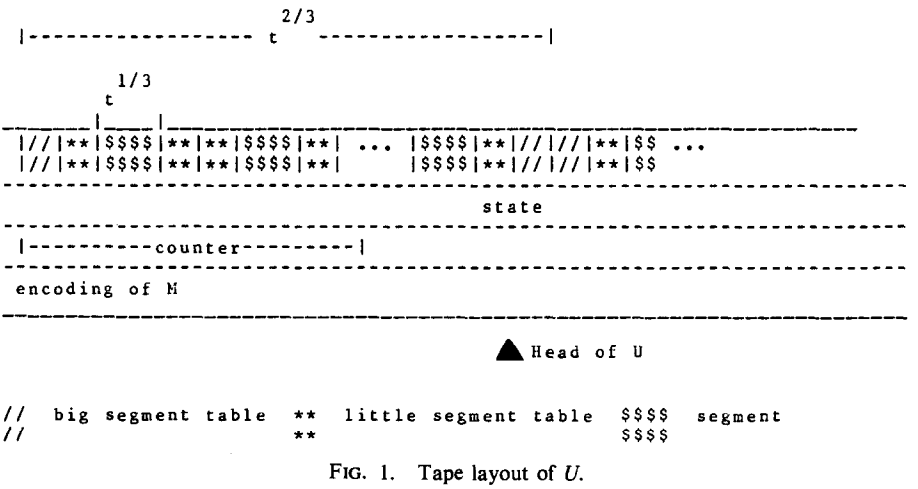

FIG. 1. Tape layout of U.

Track 4 holds an encoding of the simulated machine M, also written in the tape alphabet of U. The length of this encoding will be some constant "$m$size." To begin the simulation, U will compute segment tables for the initial contents of the work tape of M. Then U writes the initial state q_0 under the first segment table using constant ink. Now U consults the segment table:

(1) If the table indicates halting, U finishes;

(2) If the table indicates silent looping, U finishes;

(3) If the table indicates exit in state q', U writes q' under the appropriate adjacent segment table;

(4) If the table indicates printing, U performs the updating operation described below.

The first three cases require constant ink by U independent of the length of the segment; this is an important implementation point of our construction. In order to manipulate the state information and to carry out the reconfiguation described below, U may have to transfer information across long stretches of its work tape. This may require lots of head motion but the ink required is proportional to the length of the transferred string, not to the distance over which the string is transferred. Figure 2 shows how U would move the state encoding of M from one end of a segment to the

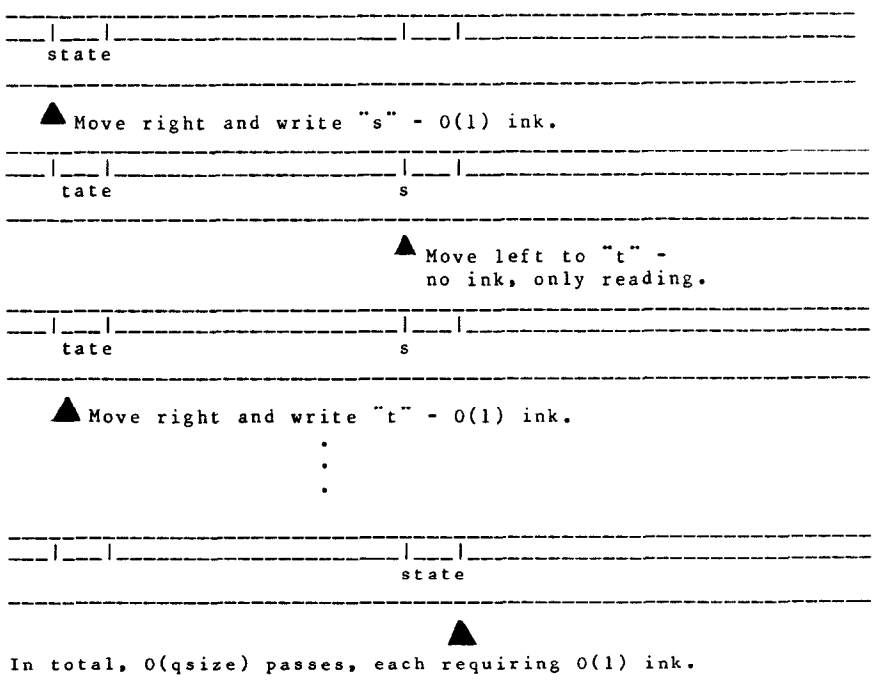

FIG. 2. Transfer operation.

other. Between successive transferred symbols the head of U must move from one end of the segment to the other but this takes no ink.

Suppose M has entered an infinite silent loop that spans segments. If M makes more that $2rt^{1/3}$ silent segment transition, U may conclude that M is looping. This bound counts r states, $t^{1/3}$ segments, and two ways to enter each segment. Here is the essential efficiency of the construction—long silent operation by M can be simulated by U in jumps of length $t^{2/3}$, each requiring only constant ink. To detect silent looping, U need only count to $O(t^{1/3})$ between printing moves by M. Since t is bounded by $f(n)$ this counting can cost at most $O(f(n)^{4/3})$ for the entire simulation.

Suppose M decides to print within a segment. The segment tables then become invalid and must be recomputed requiring as much as $O(f(n)^{2/3})$ ink. This could happen $O(f(n))$ times and could be too expensive. Here is where the level 2 segment tables come into play. If M decides to print in a big segment, U simulates the operation of M in this big segment to determine the little segment to be changed. This can be done with only $O(t^{1/3})$ ink using the tables for the little segments. The symbol is printed and the two tables for the modified little segment are recomputed using $O(t^{1/3})$ ink. Now the two tables for the containing big segment must be recomputed. However, U can now consult the up-to-date little segment tables to do this with $O(t^{1/3})$ ink rather than $O(t^{2/3})$ ink, since U can move the head of M within the segment in jumps of size $t^{1/3}$. Again, since M can print at most $f(n)$ times and t is at most $f(n)$, all this updating requires at most $O(f(n)^{4/3})$ for the entire simulation.

After a printing move the (simulated) head of M is positioned inside some little segment of length $t^{1/3}$. U continues the simulation inside of this small segment. If M specifies a printing move, U updates the segment table for this little segment then updates the tables for the containing big segment. All this takes $O(t^{1/3})$ ink as before. If M moves silently within the little segment more than $2rt^{1/3}$ steps U concludes that M is looping. If the head of M moves to the end of the small segment U may start using the little segment tables to simulate the action of M up to the next printing move. Again, M may jump across little segments at most $2rt^{1/3}$ times without printing or looping. If the head of M finally moves out to the end of a big segment U resumes the simulation at the top level.

The value of the counter on track 3 was defined as the number of silent moves by M since the last printing operation. Therefore the counter must be reset to zero whenever M prints. This is done by erasing all the tally marks and takes ink proportional to the length of the counter. The ink cost for erasing is the same as the ink cost for incrementing the counter so we may charge the cost of erasure against the cost of the incrementation and not change the asymptotic bound for the ink usage of U.

3. Reconfiguration

We have tacitly assumed above that t is not changing. However, some of the printing operations of M may increase the length of the non-blank tape contents as

simulated by U. This can be absorved by a padding segment on the end. However, if too many new segments are added without increasing the length of the segments, the number of segments may exceed $O(f(n)^{1/3})$. This would ruin the previous analysis.

Periodically during the simulation U must reconfigure the simulated tape of M. That is, the big segment and little segment tables are cleared, $t^{1/3}$ and $t^{2/3}$ are computed afresh, and new big segments and little segments are marked off. This can all be done by U using $O(t)$ ink.

Suppose the tape is reconfigured every time that it doubles in length from a previous reconfiguration. Then there can be at most $O(\log f(n)$ reconfigurations and the reconfiguring contributes an ink cost at most $O(f(n)\log f(n))$ for the entire simulation.

Right before a reconfiguration we may have a tape of length t divided into segments of length $(t/2)^{2/3}$. The number of segments is then $t/(t/2)^{2/3} = 2^{2/3}t^{2/3} = O(t^{1/3})$ and there are never too many segments.

4. The General Case

In general, for fixed $k > 1$, we can give a $(k-1)$ level construction achieving a bound of $O(f(n)^{1+1/k})$. At the ith level, U maintains segments of length $< f(n)^{(k-i)/k}$. The smallest segments at level $k-1$ are of length $< f(n)^{1/k}$. Updating the tables for one of these smallest segments costs $O(f(n)^{1/k})$ ink. Then we must work back up to the containing level 1 segment. After updating a segment of length $f(n)^{(k-i)/k}$ we must update the containing segment of length $f(n)^{(k-i+1)/k}$ but this requires only $[f(n)^{(k-i+1)/k}]/[f(n)^{(k-i)/k}] = O(f(n)^{1/k})$ ink since we can now use the up-to-date tables of the smaller segments.

During the simulation, if the head of M is not located inside of some smallest segment it is positioned at some ith level segment table.

Simulation inside a smallest segment requires $O(t^{1/k})$ ink per printing move of M. Within $2rt^{1/k}$ moves M should print or reach one end of the smallest segment; if not U may conclude that M is looping.

When U has the head of M positioned at an ith level segment table it may simulate silent operation of M in jumps of length $t^{(k-i)/k}$. Within $2rt^{1/k}$ jumps M should print or reach the end of the level i segment; if not U may conclude that M is looping.

Notice that without printing, the (simulated) head of M may only move "up" in levels. Therefore, U need count to at most $k2rt^{1/k}$ or $O(t^{1/k})$ between printing moves of M.

5. Programming Details

Complete details for the entire construction would be tedious. We only mention some points which might be unclear.

U is to reconfigure every time the simulated work tape doubles in length. After a reconfiguration has finished with a tape of length t, U should make a mark on the tape $2t$ spaces from the end. If the tape extends to this mark a reconfiguration is needed.

U must compute segment lengths which are rational powers of t. We show how to mark off $t^{3/4}$ using $O(t)$ ink. The computation for other rational powers < 1 should be clear.

U stores an integer k in unary as a string $\#^k$. U maintains four variables k, $k2$, $k3$, $k4$ satisfying:

$$k2 = k^2, \qquad k3 = k^3, \qquad k4 = k^4.$$

U executes the following loop:

$k \quad 1$
$k2 := 1$
$k3 \quad 1$
$k4 \quad 1$
while $k4 < t$ do
  $k \quad k + 1$
  $k2 := k2 + 2k + 1$
  $k3 \quad k3 + 3k2 + 3k + 1$
  $k4 \quad k4 + 4k3 + 6k2 + 4k + 1$

Addition is done by concatening strings and takes ink proportional to the length of the sum. The total ink used by the loop would be proportional to the length of the longest string formed. This is $O(t)$. When the loop finishes, k approximates $t^{1/4}$ and $k3$ approximates $t^{3/4}$.

6. Corollary

Let $\mathrm{INK}(f(n)) = \{L(M) \mid M$ *is an* $f(n)$ *ink-bounded deterministic machine*$\}$.

Define a function $g: N \to N$ *to be "ink constructable" if some deterministic machine* M *can mark off* $\#^{g(n)}$ *for inputs of length* n *using* $O(g(n))$ *ink.*

For any $\varepsilon > 0$ *if*

(1) $\inf_{n\to\infty}(f(n)^{1+\epsilon}/g(n)) = 0$,

(2) g *is ink constructable*

then there is a language in $\mathrm{INK}(g(n))$ *not in* $\mathrm{INK}(f(n))$.

Proof. Our proof uses the standard technique already developed for tape-bounded classes [3] and time bounded classes [2]. We give only a sketch.

Construct a universal machine U_g which for input of length n marks off $\#^{g(n)}$. During subsequent processing if U_g prints a symbol it also erases one $\#$. If U_g ever erases the last $\#$ it stops; this assures that U_g is $O(g(n))$ ink bounded. Now U_g decodes its input w as a machine description "M" and padding x (if the input is not of this form, U_g simply stops). U_g simulates $M(w)$ as above until:

(1) U_g runs out of ink,

(2) $M(w)$ loops silently in which case U_g accepts w,

(3) $M(w)$ halts and accepts w in which case U_g rejects w,

(4) $M(w)$ halts and rejects w in which case subg accepts w.

Now suppose $L(U_g) = L(M_0)$ where M_0 is $f(n)$ ink bounded. For some long enough padding x_0, U_g can complete the simulation of $M_0(\text{`}M_0\text{'}, x_0)$)and do the opposite. This shows that $L(U_g)$ is not computed by any $f(n)$ ink-bounded machine.

7. Extensions and Open Problems

One natural extension of this work would be to study non-deterministic ink complexity. It is also possible that our techniques could be used to improve the universal machine to $O(f(n)\log f(n))$ ink usage but this is not at all straightforward. Passing our construction to the limit where $k = O(\log f(n))$ requires storing the simulated work tape as a tree of depth $O(\log f(n))$. Now moving up and down in this tree is difficult since U cannot keep its place in the recursive structure by reading one of a fixed number of symbols. A bound of $O(f(n)\log f(n)^k)$ might be possible, again for fixed k.

Acknowledgments

We gratefully acknowledge Paul Dietz and Steve Fortune for their help with some of the details of this construction. Nick Pippenger, of IBM Research, alerted me to Ref. [4]. I would also like to thank Juris Hartmanis for posing this problem in the first place, and the referee for valuable criticism.

References

1. F. C. Hennie and R. E. Stearns, Two-tape simulation of multi-tape Turing Machines, *J. Assoc. Comput. Mach.* **13** (4) (1966), 533–546.
2. W. Paul, On time hierarchies, "Proceedings of the Ninth Annual ACM Symposium on the Theory of Computing," pp. 218–222.
3. J. Hopcroft and J. D. Ullman, "Formal Languages and their Relation to Automata," pp. 151–152 Addison–Wesley, Reading, Mass., 1969.
4. J. Gill and I. Simon, Ink, dirty-tape Turing machines and Quasi-complexity measures, in "Automata, Languages and Programming" (A. Michaelson and R. Milner, Eds.), Edinburgh Univ. Press, Edinburgh, 1976.

Printed in Belgium